NASA Contractor Report 194944

ICASE Report No. 94-55

(1)

ICASE

ADAPTIVE RELAXATION FOR THE STEADY-STATE ANALYSIS OF MARKOV CHAINS

Graham Horton

Contract NAS1-19480
June 1994

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

Adaptive Relaxation for the Steady-State Analysis of Markov Chains

Graham Horton [1]

Abstract

We consider a variant of the well-known Gauss-Seidel method for the solution of Markov chains in steady state. Whereas the standard algorithm visits each state exactly once per iteration in a pre-determined order, the alternative approach uses a dynamic strategy. A set of states to be visited is maintained which can grow and shrink as the computation progresses. In this manner, we hope to concentrate the computational work in those areas of the chain in which maximum improvement in the solution can be achieved. We consider the adaptive approach both as a solver in its own right and as a relaxation method within the multi-level algorithm. Experimental results show significant computational savings in both cases.

[1]Computer Science Department, University of Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany. Email: *graham@informatik.uni-erlangen.de.* This work was supported in part by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), M/S 132C, NASA Langley Research Center, Hampton, VA, 23681-0001.

1. Introduction

We are interested in developing efficient methods for computing steady-state solutions of large continuous-time Markov chains. In particular we are interested in new, improved algorithms for general chains with which solutions can be obtained with substantially less effort than using the standard schemes. We consider in this paper the adaptive Gauss-Seidel (AGS) method as a variant of the well-known Gauss-Seidel algorithm (GS) in the form in which it is usually implemented. Our adaptive relaxation method is based on that of Rüde [7]. Adaptive relaxation dispenses with the statically ordered processing of states in favour of a dynamic strategy. By making an appropriate choice of nodes to visit, it is hoped that computations that will have little effect on the solution can be spared and the attention be concentrated on those areas of the solution vector where the solution can most efficiently be improved.

We discuss adaptive Gauss-Seidel in two different roles. First we consider it as a solution method in its own right, i.e. as a direct alternative to the standard GS scheme. Second we consider its use as a relaxation method within the recently introduced multi-level algorithm [2]. It is shown that AGS acquires a particular meaning in this context.

In section 2 we give the problem statement and introduce some notation. In section 3 we describe and discuss the adaptive Gauss-Seidel algorithm. In section 4 we briefly state the multi-level method and show the particular meaning of AGS in this context. In section 5 results of numerical experiments are presented showing the performance of the Gauss-Seidel and multi-level algorithms both with and without the adaptive modification. It will be shown that the adaptive approach can lead to improved performance in both cases. In the final section we summarize the paper.

2. Problem Description and Aggregation Equations

Consider a Markov chain consisting of n states $s_0 \ldots s_{n-1}$. Denote the unknown vector by p, where p_i is the probability of being in state s_i.

We then have to solve the system of equations

$$Pp = 0$$

with the additional condition

$$\sum_{i=0}^{i=n-1} p_i = 1$$

Note that this equation is usually written as $\pi Q = 0$ for $\pi = p^T$ and $Q = P^T$, where Q is the infinitesimal generator matrix.

A coarser representation of the Markov chain described by matrix P may be obtained by *aggregation*. This means creating a new Markov chain described by a matrix Q with the vector of state probabilities q, each of whose N states $S_0 \ldots S_{N-1}$ is derived from a number of states of the original system. Figure 1 illustrates the situation for an eight-state Markov chain P, where states are aggregated in pairs to form a four-state coarser level system Q.

In the following we will use the terms *fine level* and *coarse level* to refer to Markov chains where the latter is obtained by aggregation from the former. The relation $s_k \in S_i$ signifies that the fine level state s_k is mapped by the aggregation operation to the coarse level state S_i.

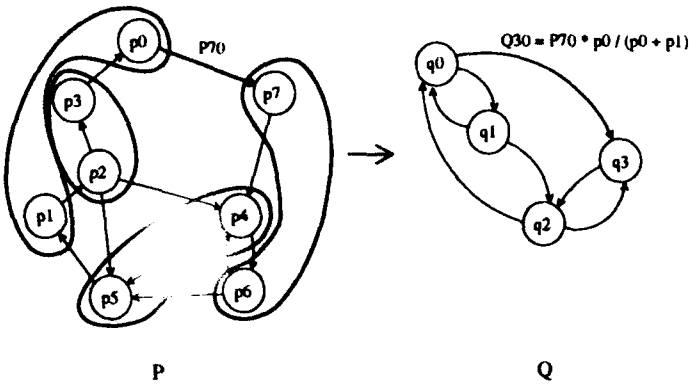

Figure 1: Aggregation of Markov Chains

The matrix Q of the aggregated system is chosen as follows :

$$Q_{ji} = \frac{\sum_{s_k \in S_i} p_k \sum_{s_l \in S_j} P_{lk}}{\sum_{s_k \in S_i} p_k} \qquad (1)$$

This is the classical *aggregation matrix*. Note that the matrix Q is a function not only of the fine level matrix P, but also of the fine level solution vector p.

This yields the aggregated equation in the unknown q:

$$Qq = 0$$
$$\sum_{i=0}^{N-1} q_i = 1 \quad .$$

It can then be shown that

$$q_i = \sum_{s_k \in S_i} p_k \quad ,$$

i.e. the solution q of the aggregated system truly represents a coarser version of the solution p of the original problem. The probability of being in state q_i is the sum of the probabilities of being in any of its constituent fine-level states. We use the aggregation equation as a basis for the multi-level method, whereby we approximate the exact solution values p_k in (1) above by values from the current iterate.

3. Adaptive Relaxation

Gauss-Seidel is an iterative method for the solution of linear systems of equations which proceeds by successively solving the local equation for each unknown by modifying the value of the unknown to make its residual equal to zero. The Gauss-Seidel method is given in figure 2. The standard Gauss-Seidel method visits each state exactly once per iteration step. In addition the order in which states are visited is fixed and in practice is determined largely coincidentally by the order in which states have been generated. In the case of Markov chains derived from generalized stochastic Petri-nets (GSPNs) the ordering is a depth-first or breadth-first traversal starting at the state representing the initial marking.

We may consider the *effectiveness* of the Gauss-Seidel method at any one state s_i at any time during the computation to be the amount by which the current solution value $\bar{p}_i$ changes when

the GS relaxation step is applied to that state only. Thus GS is effective when it is able to bring about a large improvement in the state's value, and is ineffective otherwise. However, if the method is converging towards the steady-state solution, then changes in the solution must become successively smaller. Thus the effectiveness is a relative measure, meaningful only with respect to the effectiveness at other states of the chain at a given instant during the computation.

Intuition tells us that the effectiveness of GS at any point in time during the solution process can vary greatly between different states. In the Markov chain below, for example, there are many states for which the effectiveness is initially zero, i.e. application of GS at these states would not change the solution at all! Ideally, of course, we would like to apply GS to those states where the effectiveness is highest. In this case, the computational effort would be expended with maximum efficiency. Conversely we would like be able to pass over states with low effectiveness and not spend any computational effort on them. As the computation progresses, the effectiveness of states changes as the values of their neighbours are modified. Unfortunately, of course, we do not know which states have the highest effectiveness, i.e. the largest residuals, and finding them would essentially involve performing GS at every state, thus destroying the very advantage we were hoping to achieve.

We must therefore adopt a different strategy, which is derived from that of Rüde [7], who applied adaptive smoothing to the solution of partial differential equations. We introduce a set of states, called the *active set*, which is an approximation to the set of states with high effectiveness at the current stage of the computation. By only considering states from this set for the application of GS, we hope to concentrate our computational effort in the "hot spots" - those areas in which GS is able to achieve substantial improvements to the solution.

The adaptive Gauss-Seidel algorithm is given in figure 3, where M denotes the active set of states. Since we do not know initially which are the states with high effectiveness, we are forced to initialize the active set M to include all states in the Markov chain. The main loop of the algorithm repeats until M has become empty. A state s_i to be relaxed is chosen and removed from M and its current solution value stored in a temporary variable t. The relaxation is then performed. If the change in the solution value exceeds a pre-defined limit ϵ then all states in the chain that are directly influenced by state s_i are inserted into M. The motivation for this set update strategy is that a large change in state s_i changes the residual by a proportionate amount at those states whose values depend on $\bar{p}_i$, and thus it is likely that a high effectiveness is induced there.

Upon termination of AGS, we assume that solving the local equation at any state cannot improve the solution there by more than an amount proportional to ϵ. Although this means that little improvement can be achieved locally, this of course tells us nothing about the accuracy of the solution. We must therefore repeat the procedure with a reduced value of ϵ. We choose a simple strategy given by the following algorithm:

```
procedure solve_with_AGS
    ε = ε0
    while ||Pp̄||∞ > δ do
        adaptive_Gauss_Seidel(ε)
        ε = ε * Δε
```

with $\Delta\epsilon$ chosen to satisfy $0 < \Delta\epsilon < 1$.

As a simple example we consider solving the birth/death chain of figure 4 with a length of $n = 33$, a birth rate of $\lambda = 49$ and a death rate of $\mu = 50$. Setting $\rho = \lambda/\mu$ and counting from $i = 0$, this chain has the solution

$$p_i = \frac{1-\rho}{1-\rho^n}\rho^i \quad .$$

If, as is typically the case in practice, we initialize the solution vector to the constant function $(\frac{1}{n}, \ldots, \frac{1}{n})$, then we can observe that the application of GS to any state other than s_0 and s_{n-1} will have no effect, i.e. the computational effort would be completely wasted. Only at the ends of the chain can an improvement be achieved initially. This is because the initial guess solves the local equations at every internal state:

$$\frac{1}{\lambda+\mu}\left(\lambda\frac{1}{n} + \mu\frac{1}{n}\right) = \frac{1}{n} \quad .$$

We show the history of the active set in figure 5, where the nodes of the chain are plotted vertically and time horizontally. A cutoff value of $\epsilon = 1e-4$ was used. An "X" denotes a state currently in the active set and a "." a state not currently in the active set, whilst an "O" shows the state currently being relaxed. During the computation, each node is initially visited once, as all nodes start out in the active set. However, it can be seen that only at the ends of the chain are nodes put back into the set and the computation proceeds to treat only nodes $0 \ldots 4$ and $n-5 \ldots n-1$. It is important to realize that this would hold regardless of the size of the chain, thus the proportion of nodes that would be visited after the initial sweep can be extremely small. Hence this method achieves the required result: it only expends computational effort in those portions of the chain in which substantial improvements can be achieved. When the active set has emptied, ϵ is reduced and the adaptive procedure repeated with the smaller tolerance. Note that we initialize the active set to include all states of the chain in order to err on the side of safety; in this particular example we could have achieved essentially the same result at significantly reduced cost by initializing the active set to just $\{s_0, s_{n-1}\}$.

4. Multi-Level Solution Algorithm

In this section we briefly review the recently introduced multi-level algorithm, details of which can be found in [2]. The multi-level algorithm is based on a recursive aggregation of the Markov chain to obtain approximations of successively smaller dimensions. The algorithm passes through all levels of the hierarchy of chains in a downward-upward sweep. Solutions on finer levels are used to construct coarser equations, the approximate solutions of which are used to correct those on the next finest level. The coarser level equations are the aggregation equations of section 2.

We adopt the following abbreviations for vectors $a,\ b,\ c \in \mathbb{R}^m$:

$$\begin{aligned} a = b * c &\equiv a_i = b_i * c_i, \quad 1 \le i \le m \\ a = b/c &\equiv a_i = b_i/c_i, \quad 1 \le i \le m \end{aligned}$$

The *two*-level version of the ML iteration is given by the following sequence of steps.

- Perform GS relaxation on finer level

$$\tilde{p} = GS(p^{(i)})$$

- Restrict solution to coarse level

$$\tilde{q} = R(\tilde{p}) \quad \equiv \quad \tilde{q}_i = \sum_{s_k \in S_i} \tilde{p}_k$$

- Compute coarse level aggregation matrix

$$\tilde{Q}_{ji} = \frac{\sum_{s_k \in S_i} \tilde{p}_k \sum_{s_l \in S_j} P_{lk}}{\tilde{q}_i} \tag{2}$$

- Solve coarse equation for q

$$\tilde{Q}q = 0 \ , \quad \sum_{i=0}^{N-1} q_i = 1 \tag{3}$$

- Compute coarse level correction

$$q^* = \bar{q}/\tilde{q}$$

- Compute fine level correction

$$p^* = I(q^*) \quad \equiv \quad p_k^* = q_i^*$$

- Apply fine level correction

$$p^{(i+1)} = \bar{p} = C(\tilde{p}, p^*) \quad \equiv \quad \tilde{p} * p^*$$

In this two-level form the method is similar to well-known iterative aggregation/disaggregation (IAD) methods such as those of Koury, McAllister and Stewart [3] and of Takahashi [9]. The *multi*-level algorithm is obtained by recursive application of the two-level algorithm to obtain a solution to the aggregated equation (3) and is described in algorithmic form in figure 6. We use the subscript l to denote level of representation ($l = lmax$ finest level, $l = 0$ coarsest level). The coarse level $l-1$ and fine level l between which the operators I and R map are identified by appropriate indices. Note that, because of the recursive nature of the algorithm, the unknowns q^*, $\bar{q}$ and $\tilde{q}$ are represented by the variables p_{l-1}^*, $\bar{p}_{l-1}$ and $\tilde{p}_{l-1}$, respectively. We allow in general the possibility of applying GS ν times at each level with $\nu \geq 1$, denoted by $\mathtt{GS}^\nu$.

The aggregation strategy used at present is very simple. It attempts to map pairs of fine level states that are strongly coupled to a common coarse level state. To achieve this, we loop through all states of a given level and for each state s_i that has not yet been assigned to an aggregated state, we choose the unassigned neighbour s_j for which P_{ji} is maximal. Any states that remain unaggregated by this policy are mapped to an assigned neighbour s_j for which P_{ji} is maximal. Thus aggregated states are almost always composed of fine-level states that are neighbours and states with the strongest coupling coefficients are aggregated together.

Adaptive Gauss-Seidel has a particular relevance when used as the relaxation component of the ML algorithm. In order to do this we borrow concepts from the multigrid literature, in particular [8]. We consider the error e_i in the current solution value $\bar{p}_i$ in state s_i

$$e_i = p_i - \bar{p}_i \quad ,$$

and differentiate between high and low frequency error components, whereby the highest frequency errors are those that oscillate in size between neighbouring states. Low frequency errors are those that vary only slowly across the chain. Upon completion of AGS, we can assume that the magnitude of high frequency error components is small everywhere, since no substantial improvements in the solution can be made locally. Thus we can conclude that the *relative* magnitudes of all unknowns

with respect to their immediate neighbours are approximately correct, regardless of the *absolute* size of the error e in those unknowns.

Consider now the definition of the coarse level matrix in the ML method (2):

$$\tilde{Q}_{ji} = \frac{\sum_{s_k \in S_i} \tilde{p}_k \left(\sum_{s_l \in S_j} P_{kl} \right)}{\sum_{s_k \in S_i} \tilde{p}_k}$$

The matrix $\tilde{Q}$ is an approximation to the correct coarse system matrix Q, which is obtained by setting $\tilde{p} = p$ above. Since approximate solutions to the coarse system are used to derive corrections to the solution at the next highest level, it is clear that the difference between $\tilde{Q}$ and Q may affect the behaviour of the ML method. In particular, if $\tilde{Q}$ is a bad approximation, then the coarse level correction may be extremely inaccurate. We surmise that it is cases such as this which have led to reports by some authors of divergence of iterative aggregation-disaggregation methods such as that of Takahashi [9] for some problems.

The quality of $\tilde{Q}$, i.e. the size of $Q - \tilde{Q}$, depends on the quantities

$$\frac{\tilde{p}_k}{\sum_{s_k \in S_i} \tilde{p}_k} \quad ,$$

which is the conditional probability of the Markov chain being in state s_k, given that it is in the aggregate state S_I. Thus it is not necessary to know the absolute size of $\tilde{p}$ in order to be able to compute a correct value for $\tilde{Q}$, but it is sufficient to know the relative sizes of all fine-level unknowns which are aggregated to a common coarse level state. This set of fine-level states is by the definition of the aggregation strategy always composed of close neighbours and in most cases is a subset of the set of all immediate neighbours of any state. For any fine-level state s_j and coarse level state S_i for which holds $s_j \in S_i$ we have as a rule

$$\{s_k : s_k \in S_i\} \subseteq \{s_l : P_{jl} \neq 0\} \quad .$$

Thus AGS gives us a means to control the quality of the coarse level matrix by eliminating high-frequency errors to a controlled tolerance at a possibly greatly reduced cost compared to GS.

5. Experimental Results

Figure 7 shows a multiprocessor system in which the n processors $Pr_1 \ldots Pr_n$ compete for access to two memory units CM1, CM2 via a common bus. Marsan, Balbo and Conte [4] give a GSPN model of this multiprocessor (the structure of which is shown in figure 8) which allows for the possibility of failure and repair of the processors, the bus and the memory units. The model allows the computation of the effective utilization of the processors in the presence of failures and competition for the system resources.

Figure 9 shows the computational work of the CS, AGS, ML-GS and ML-AGS methods applied to this problem, where ML-GS (ML-AGS) denotes the multi-level method using standard (adaptive) Gauss-Seidel as a relaxation component. We show the total number of millions of floating point operations used as a function of problem size measured as the number of processors in the model. The number of states of the Markov chains varied from 91 (2 processors) to 3883 (10 processors). All problems were solved to an accuracy of $\|P\bar{p}\|_\infty < 1e-9$.

Comparing GS and AGS, we see that we are able to achieve a substantial improvement via the adaptive strategy. For the smallest problem considered, AGS is a factor of about 3.6 faster (not discernible in the figure); for the largest it is about 9.6 times faster.

In order to compare ML-GS and ML-AGS we magnify the lower section of figure 9, shown as figure 10. Here we see that the adaptive technique also improves the multi-level method. Since, however, the ML-GS method is already very efficient for this problem, needing only between eight and ten iterations to achieve convergence, there was little room left for improvement for ML-AGS. Both ML schemes are still substantially faster than AGS.

Comparing the standard GS and ML-GS schemes, we see that although these are problems of very small size, the saving in computational effort of ML over GS is quite dramatic: a factor of 39 for the smallest and of 77 for the largest problems considered. It is also clear that the gap widens as the problem size is increased. It is results such as these, see [2] for more examples, that make us confident that the multi-level method is a strong candidate as a steady-state solver for Markov chains.

Figure 11 shows the results of the four methods applied to the example stochastic Petri-net in the original paper of Molloy [5], the structure of which is shown in figure 12. For this problem, the computational work of GS grows extremely fast with problem size, measured by the number of tokens k in the initial marking. The number of states of the Markov chain varies from 506 to 23821. AGS is a distinct improvement, being already factors of 5 and 8 faster for 20 and 30 tokens respectively. ML-GS is about twice as fast as AGS throughout and ML-AGS another factor of 2 to 4 faster still. Thus the overall improvement from the standard scheme to the best new scheme is a factor of 40 for 30 tokens and increases as the problem grows larger.

Figure 13 shows the results of the four methods applied to a model of a processor cluster with failures and repairs by Muppala and Trivedi [6] (figure 14). In the model jobs arriving can be processed or rejected, depending on whether the system is down or up. A quorum of active processors can be specified, which determines whether jobs can be accepted by the system or not. Enabling functions (not shown) are used to define the model's behaviour. In addition, the size of the buffer receiving the jobs can be varied. We chose to scale the size of the problem by varying the buffer length between 8 and 64, yielding Markov chains with 81 ... 585 states. For this model, the operation count for GS grows sharply, but linearly with buffer size, whereas the other methods only grow at a more modest rate. ML-AGS is superior to ML-GS by approximately 30% throughout; both are about four times faster than AGS, despite the fact that this is an extremely small problem.

6. Conclusions

In this paper we have described and discussed the adaptive Gauss-Seidel method as a variation of the well-known Gauss-Seidel solver for Markov chains. We also gave a brief description of the multi-level algorithm which was recently introduced in [2] and which has been shown to often require significantly less computation time than the standard scheme for a number of test problems. Experimental results showed that the introduction of an adaptive strategy can improve the performance of the Gauss-Seidel method by almost an order of magnitude, and that it can also be used to advantage as a component of the ML algorithm.

Possible extensions and modifications are to take the coefficients P_{ij} into account when deciding whether neighbouring states are to be inserted into the active set. This would more accurately reflect the change in magnitude of the residual in those states, which would avoid insertion of states with low effectiveness and thus lead to further computational savings. One might also consider

an adaptive SOR scheme, in which the basic AGS method is modified to allow overrelaxation. A problem which we have yet to resolve satisfactorily is the automated choice of values for the parameters ϵ_0 and $\Delta\epsilon$ in the AGS scheme. Alternatively, one might consider a dynamic tuning of ϵ_0 during the computation.

Further work will include the implementation of a "fully adaptive" multi-level solver in a manner similar to Rüde [7] - one in which the active set processing crosses the levels of the hierarchy. In this way it is hoped that an adaptive relaxation which remains restricted to local areas of the chain on one level initializes the active set on neighbouring levels to a corresponding subset of the states on those levels. This may lead to substantial savings for the ML-AGS method, as restarting each level with a full active set could be avoided.

Acknowledgements

The Markov chains were generated using the SPNP software package of G. Ciardo [1], who also supplied the GSPN models for the second and third test problems.

References

[1] G. CIARDO, K. TRIVEDI, J. MUPPALA: *SPNP: stochastic Petri net package.* Proc. of the Third Int. Workshop on Petri Nets and Performance Models (PNPM89), Kyoto, Japan, pp. 142-151 Dec, 1989. IEEE Computer Society Press.

[2] G. HORTON, S. LEUTENEGGER: *A Multilevel Solution Algorithm for Steady-State Markov Chains.* ICASE Report #93-81 NASA CR-191558, NASA Langley Resarch Center, September 1993. *And* SIGMETRICS 94 (Proceedings), Nashville, TN, 16th-20th May, 1994.

[3] R. KOURY, D. MCALLISTER, W. STEWART: *Methods for computing stationary distributions of nearly completely decomposable Markov chains..* SIAM J. Alg. Disc. Math. 5, 2 (1984), 164-186.

[4] M. MARSAN, G. BALBO, G. CONTE : Performance models of multiprocessor systems. MIT Press, 1988.

[5] M. MOLLOY: *Performance analysis using stochastic Petri nets.* IEEE Trans. Comp. Vol 31 No. 9, 913-917, Sept. 1982.

[6] J. MUPPALA, K. TRIVEDI: *GSPN Models: Sensitivity Analysis and Applications.* Proc. of the 28th ACM Southeast Region Conf., Greenville, SC, 1990, 24-33.

[7] U. RÜDE: *On the Multilevel Adaptive Iterative Method.* Technical Report TUM-I9216, Computer Science Dept., Technische Universität München, 1992.

[8] J. RUGE, K. STÜBEN: *Algebraic Multigrid.* In S. McCormick (Ed.) *Multigrid Methods.* SIAM, 1987.

[9] Y. TAKAHASHI: *A Lumping Method for Numerical Calculations of Stationary Distributions of Markov Chains,* Research Report No. B-18, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1975.

```
procedure standard_Gauss_Seidel
    for i = 0 to n − 1 do
        p̄_i ← (1/P_ii) Σ_{j≠i} P_ij p̄_j
```

Figure 2: Standard Gauss-Seidel Algorithm

```
procedure Adaptive_Gauss_Seidel(ε)
    M = {s_1 ... s_n}
    while M ≠ ∅
        choose state s_i ∈ M
        M = M \ s_i
        t = p̄_i
        p̄_i ← (1/P_ii) Σ_{j≠i} P_ji p̄_j
        if |t − p̄_i| > ε
            for all j ≠ i, P_ji ≠ 0
                M = M ∪ s_j
```

Figure 3: Adaptive Gauss-Seidel algorithm

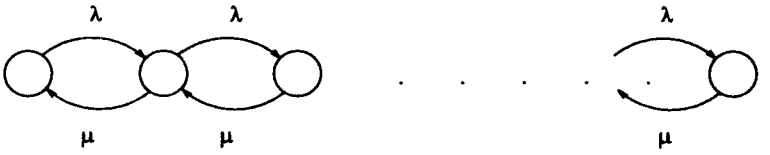

Figure 4: Birth-Death Markov Chain

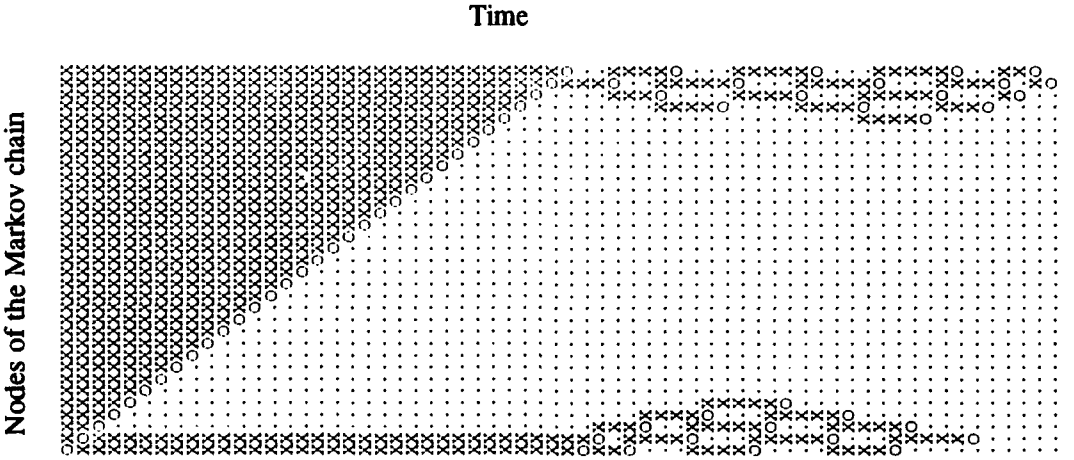

Figure 5: Active set history

```
procedure ml(l)
    if (l = 0)
        solve P_l p̄_l = 0
    else
        p̃_l = GS^ν(p̄_l)
        p̃_{l-1} = R_{l-1,l}(p̃_l)
        ml(l − 1)
        p*_{l-1} = p̄_{l-1}/p̃_{l-1}
        p*_l = I_{l-1,l}(p*_{l-1})
        p̄_l = C(p̃_l, p*_l)
    return
```

Figure 6: Multi-Level Algorithm

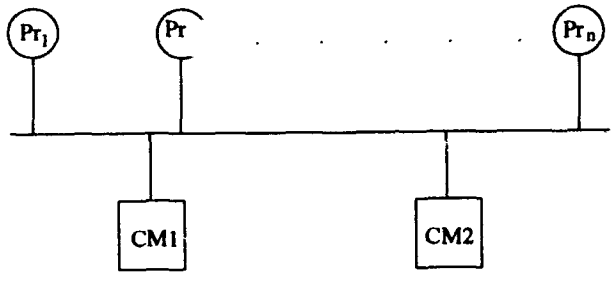

Figure 7: Simple Multiprocessor Example

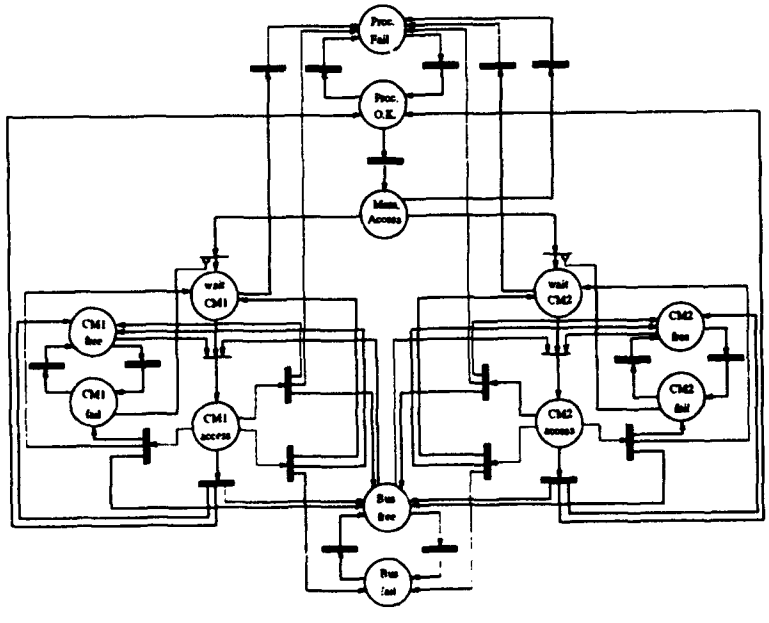

Figure 8: Marsan/Balbo/Conte Multiprocessor Model

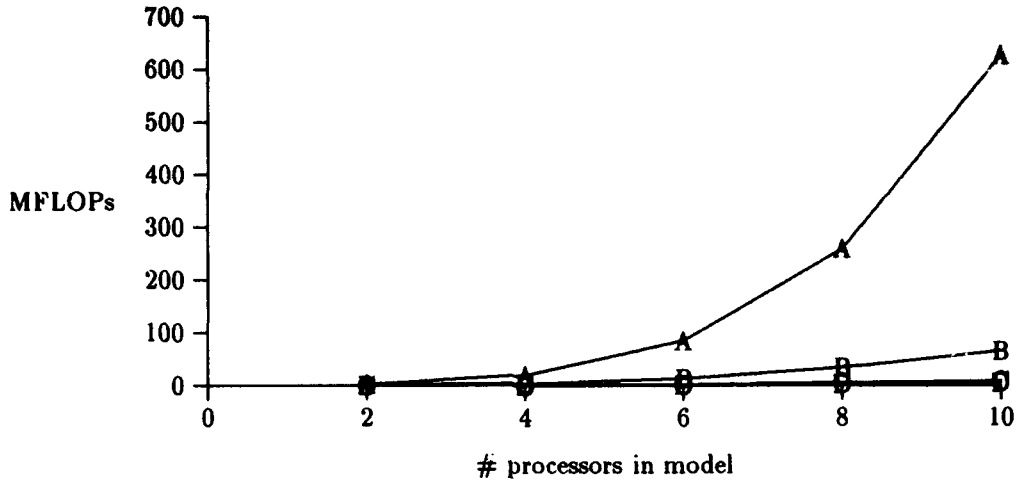

Figure 9: Computational work for GS (A), AGS (B), ML-GS (C) and ML-AGS (D) to solve the Marsan/Balbo/Conte problem.

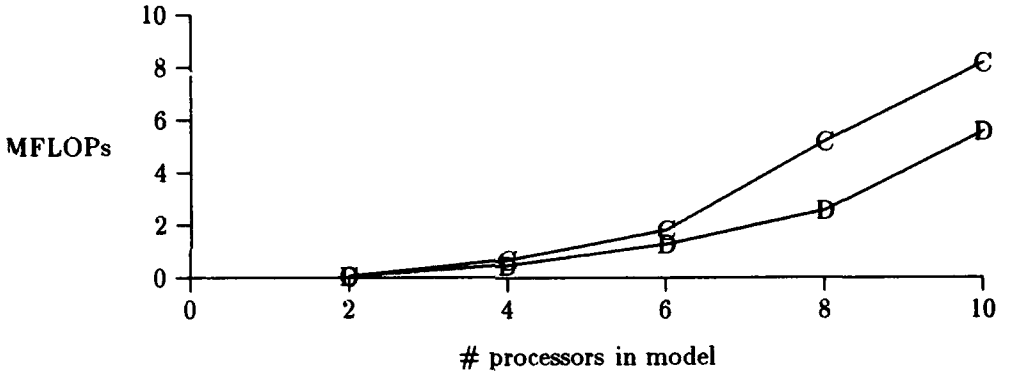

Figure 10: Computational work for ML-GS (C) and ML-AGS (D) to solve the Marsan-Balbo-Conte problem (Magnification of part of figure 9).

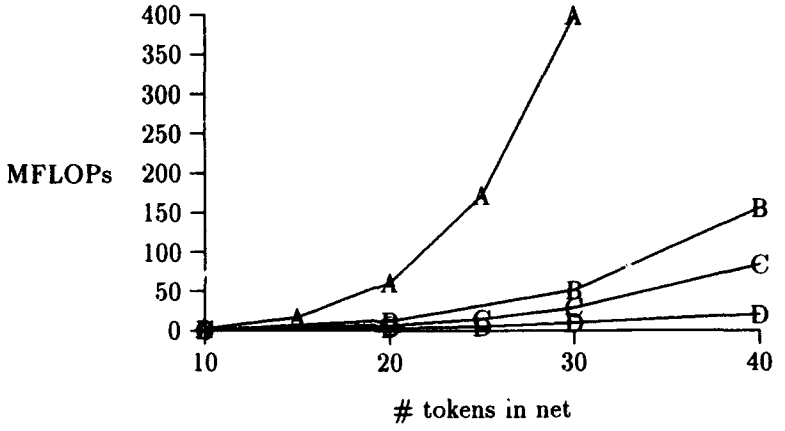

Figure 11: Computational work for GS (A), AGS (B), ML-GS (C) and ML-AGS (D) to solve Molloy's example SPN.

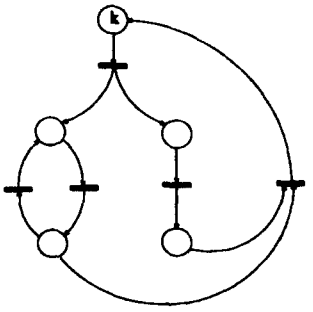

Figure 12: Molloy's example SPN.

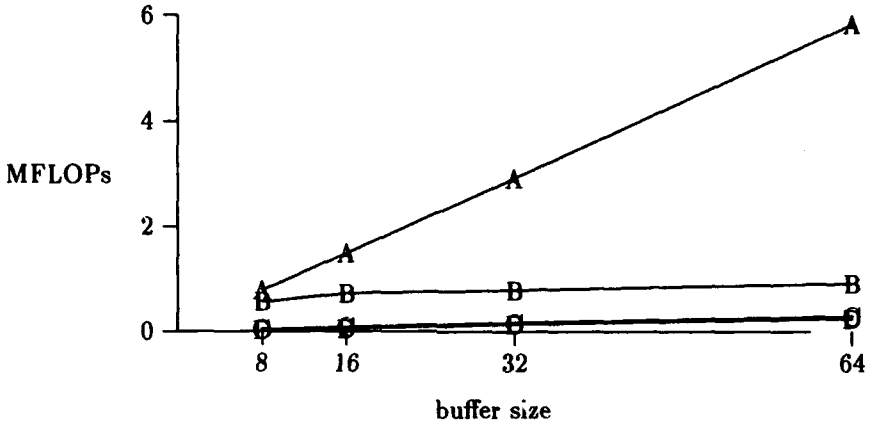

Figure 13: Computational work for GS (A), AGS (B), ML-GS (C) and ML-AGS (D) to solve the Muppala-Trivedi model.

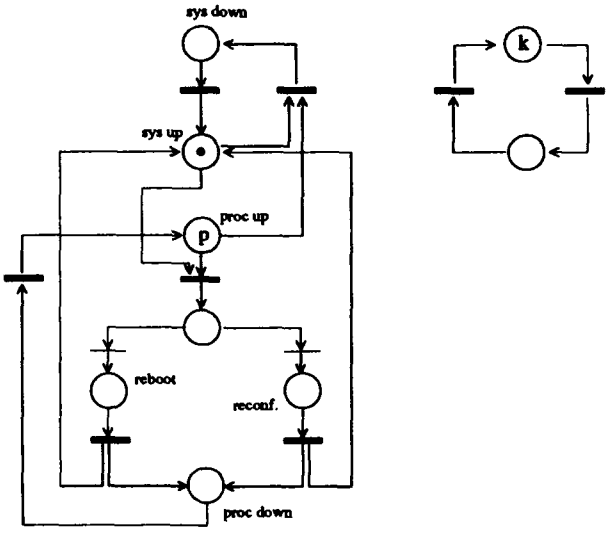

Figure 14: GSPN of Muppala-Trivedi

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1994	Contractor Report

4. TITLE AND SUBTITLE
ADAPTIVE RELAXATION FOR THE STEADY-STATE ANALYSIS OF MARKOV CHAINS

5. FUNDING NUMBERS
C NAS1-19480
WU 505-90-52-01

6. AUTHOR(S)
Graham Horton

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science
and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 94-55

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-194944
ICASE Report No. 94-55

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report
Submitted to Numerical Solutions of Markov Chains, Conference

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We consider a variant of the well-known Gauss-Seidel method for the solution of Markov chains in steady state. Whereas the standard algorithm visits each state exactly once per iteration in a pre-determined order, the alternative approach uses a dynamic strategy. A set of states to be visited is maintained which can grow and shrink as the computation progresses. In this manner, we hope to concentrate the computational work in those areas of the chain in which maximum improvement in the solution can be achieved. We consider the adaptive approach both as a solver in its own right and as a relaxation method within the multi-level algorithm. Experimental results show significant computational savings in both cases.

14. SUBJECT TERMS
multi-level methods, Markov chain, performance analysis

15. NUMBER OF PAGES
14

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

*U.S. GOVERNMENT PRINTING OFFICE: 1994-528-064/23010 Region 3.

Standard Form 298(Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102